Ubiquitous and Mobile Computing
CS 528: Visage: A Face Interpretation Engine for Smartphone Applications

Amogh Raghunath
Computer Science Dept.
Worcester Polytechnic Institute (WPI)

Mateus Amarante Araujo
Robotics Engineering Dept.
Worcester Polytechnic Institute (WPI)
Outline

- Introduction
- Related Work
- Design
- Architecture
- Implementation
- Evaluation
- Visage Applications
- Conclusion
Introduction

- Smart phones are embedded with sensors
- Users are increasingly using applications
  - Tweeting, Web surfing, texting
- Camera, capable of observing users as they interact with different application
Introduction

- **Visage**: A robust, real-time face interpretation engine for smart phones
  - Tracking user’s 3D head poses & facial expression
  - Fuse data from front-facing camera & motion sensor
Related Work

- Involves limited image processing
  - SenseCam
  - Recognizr
  - MoVi
- Simple tracking of 2D face representations
  - PEYE
- **Visage**: A robust, real-time face interpretation engine for smart phones
Design

Challenges

- User Mobility
  - Movement of the phone cause low image quality
  - Analyze exposure level of face region

- Limited Phone Resources
  - Operate in real-time
Architecture

Sensing Stage
- Camera
- Motion sensor

Preprocessing Stage
- Adaptive exposure
- Face detection
- Phone posture

Tracking Stage
- Feature points tracking
- AAM face fitting
- Expression classification
- Pose inference

Inference Stage
Architecture

Sensing Stage

- Captures the video stream from the phone’s front-facing camera
- Raw motion data from accelerometer and gyro sensors on the phone.
Architecture

Preprocessing Stage

1. Phone posture component
2. Face detection with tilt compensation
3. Adaptive exposure component
Architecture

Preprocessing Stage

Phone posture component

- Identifies frames which contain user’s face and monitors the phone posture
- Raw readings from accelerometer and gyroscope and estimates of direction of gravity
- Calculates mean and variance on each direction
- Gravity direction – mean of accelerometer data
Architecture

Preprocessing Stage

Face detection with tilt compensation

- AdaBoost object detector with tilt correction

\[ \theta_g = \frac{180}{\pi} \arctan \frac{a_x}{a_y} \]

- Image is tilted by:

\[ I_r = \begin{bmatrix} \cos \theta_g & -\sin \theta_g \\ \sin \theta_g & \cos \theta_g \end{bmatrix} I_i \]
Architecture

Preprocessing Stage

Adaptive exposure component

- A clear face region is critical for tracking and inference
- Visage uses the local lighting information within the detected face region to correct the camera hardware exposure level.
- Exposure level by computing the centroid of $H_{face}$:

$$C_{H_{face}} = \frac{\sum_{i=0}^{255} i H_{face}(i)}{\sum_{i=0}^{255} i}$$
Architecture

Preprocessing Stage

Adaptive exposure component
Architecture

Tracking Stage

- Feature Points Tracking Component
  - Select feature points (e.g. eye corners and edges of mouth): they are stable across frames
Methodology

Tracking Stage

- Feature Points Tracking Component
  - Lucas-Kanade (LK) tracking algorithm
  - CAMSHIFT
Methodology

Tracking Stage

- Pose Estimation Component
  - Pose from Orthography and Scaling with ITerations (POSIT) algorithm
  - 4 points in Image (2D) -> 3D pose estimation
  - Human head simplified to a rigid cylinder
Methodology

Inference Stage

- **Active Appearance Model**
  - Generate appearance features for classification
  - Combine shape and texture models (more accurate)

- **Expression Classification**
  - Fisher Linear Discriminant Analysis (Fisherface) to reduce the dimension of the face feature vector
  - Support Vector Machine (SVM) classifier with LibSVM
Implementation

- iPhone 4
- OpenCV Library
- AAM from VOSM (Vision Open Statistical Models)

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 × 480</td>
<td>4090</td>
</tr>
<tr>
<td>480 × 360</td>
<td>2123</td>
</tr>
<tr>
<td>320 × 240</td>
<td>868</td>
</tr>
<tr>
<td>192 × 144</td>
<td>298</td>
</tr>
<tr>
<td>160 × 120</td>
<td>203</td>
</tr>
<tr>
<td>96 × 72</td>
<td>68</td>
</tr>
<tr>
<td>80 × 60</td>
<td>53</td>
</tr>
</tbody>
</table>
Evaluation

● Benchmarks

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Avg. CPU usage</th>
<th>Avg. memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI only</td>
<td>&lt; 1%</td>
<td>3.18MB</td>
</tr>
<tr>
<td>Pose estimation</td>
<td>58%</td>
<td>6.07MB</td>
</tr>
<tr>
<td>Expression inference</td>
<td>29%</td>
<td>4.57MB</td>
</tr>
<tr>
<td>Pose estimation &amp; expression inference</td>
<td>68%</td>
<td>6.28MB</td>
</tr>
</tbody>
</table>

Table 2. CPU and memory usage under various task benchmarks

<table>
<thead>
<tr>
<th>Component</th>
<th>Average processing time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face detection</td>
<td>53</td>
</tr>
<tr>
<td>Feature points tracking</td>
<td>32</td>
</tr>
<tr>
<td>AAM fitting</td>
<td>92</td>
</tr>
<tr>
<td>Facial expression classification</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3. Processing time benchmarks
Evaluation

- Tilted Face Detection

  Standard AdaBoost face detector vs Visage’s Detector
Evaluation

- Motion Based Reinicialization

Reinitialize when variance is high
Evaluation

- Head Pose Estimation

Fig. 8. Head pose estimation error
Evaluation

- Facial Expression Classification
  - Validation with The Japanese Female Facial Expression (JAFFE) Database

<table>
<thead>
<tr>
<th>Expressions</th>
<th>Anger</th>
<th>Disgust</th>
<th>Fear</th>
<th>Happy</th>
<th>Neutral</th>
<th>Sadness</th>
<th>Surprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy(%)</td>
<td>82.16</td>
<td>79.68</td>
<td>83.57</td>
<td>90.30</td>
<td>89.93</td>
<td>73.24</td>
<td>87.52</td>
</tr>
</tbody>
</table>

Confusion Matrix
Applications

- Streetview+

(a) Streetview+ on the go  (b) Head facing front
(c) Head facing left  (d) Head facing right
Applications

- Mood Profiler
Conclusion

- Face-aware applications
- Designed for resource limited mobile phones
- Online processing at a lower computational cost
- Multi-modality sensing
- Flexible and robust
References

- [http://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf](http://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf) (AdaBoost)
- [http://docs.opencv.org/master/d7/d8b/tutorial_py_lucas_kanade.html#qsc.tab=0](http://docs.opencv.org/master/d7/d8b/tutorial_py_lucas_kanade.html#qsc.tab=0) (LK tracking algorithm)
- [http://docs.opencv.org/master/db/df8/tutorial_py_meanshift.html#qsc.tab=0](http://docs.opencv.org/master/db/df8/tutorial_py_meanshift.html#qsc.tab=0) (CAMSHIFT)
- [http://www2.imm.dtu.dk/~aam/main/](http://www2.imm.dtu.dk/~aam/main/) (AAM algorithm)
- [http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html](http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html) (SVM algorithm)
- [http://www.csie.ntu.edu.tw/~cjlin/libsvm/](http://www.csie.ntu.edu.tw/~cjlin/libsvm/) (LibSVM)
- [http://www.kasrl.org/jaffe.html](http://www.kasrl.org/jaffe.html) (JAAFE Database)
Thank you!